Available online at www.sciencedirect.com

ScienceDirect

Procedia Technology 26 (2016) 436 – 446

Procedia Technology

3rd International Conference on System-integrated Intelligence: New Challenges for Product and Production Engineering, SysInt 2016

Approach for an Integrated Model-Based Design of Intelligent Dynamic Systems Using Solution and System Knowledge

Felix Oestersötebier, Farisoroosh Abrishamchian*, Christopher Lankeit, Viktor Just, Ansgar Trächtler

Heinz Nixdorf Institut, University of Paderborn, Fürstenallee 11, 33102 Paderborn, Germany

Abstract

In the design process of intelligent technical systems, simultaneous and concurrent engineering is generally encouraged on the one hand, while on the other hand cooperation and coordination of the involved disciplines is required. In multidisciplinary system development, 1) a common understanding of the objective is of vital importance for the system's success and 2) the combined artifacts of the different disciplines need to be analyzed before the system is built. In this paper, we address these issues and present an approach for an integrated model-based design process, which facilitates the use of solution and system knowledge and reduces the huge effort for building and maintaining the required simulation models. To illustrate this, we use the application example of two cooperating delta robots.

The system knowledge constitutes the basis for concurrent design in the involved disciplines, all of which provide and expand certain aspects. To analyze the domain-specific dynamic behavior of the subsystems and their components, multiple dynamic behavior models are developed in different levels of detail and domains. However, in order to analyze the complex interactions and dependencies between them, integrated models of the whole system are needed, which fit the varying modeling objectives and analysis goals. Certainly, the manual process of building up such models and maintaining consistency between all the artifacts entails great effort. To improve this, we present the concept of a Multifunctional Model Client (MMC).

Peer-review under responsibility of the organizing committee of SysInt 2016

Keywords: model-based design; discipline-spanning coordination; technical requirements; Multifunctional Model Client

1. Introduction and Problem Analysis

The design process of intelligent dynamic systems is typically characterized by close collaborations and many cross-dependencies of different domains/disciplines. Searching for new and innovative solutions and in order to improve the functionality, designers aim at using synergy effects. However, modern cyber-physical systems (CPS) not only require many of these interactions but are also expected to feature high dynamics.

* Corresponding author. Tel.: +49 5251 60-6286 ; fax: +49 5251 60-6297.
E-mail address: farisoroosh.abrishamchian@hni.uni-paderborn.de

Peer-review under responsibility of the organizing committee of SysInt 2016
doi:10.1016/j.protcy.2016.08.056

Thus, the increasing complexity can no longer be met with the current design methods very often. This is due to many challenges that arise from the lack of methods, processes, information or tools. In this paper we attempt to address aspects of the following subset (see also [1]):

- clarification of goals and requirements,
- early model-based testing of required characteristics,
- exchange of design models and data,
- cooperative work and information exchange among the design engineers,
- simultaneous consideration of designs from different disciplines,
- lack of tools and methods supporting multi-disciplinary design.

To cope with these issues, we 1) use and connect models of many different kinds, 2) re-use solutions of former developments and 3) make system knowledge available for the designers. At first, we use on the specification technique CONSENS [2] and the SysML4CONSENS profile [3] to build a discipline-spanning system model that formalizes a conjoint view on the system. Here, we focus on obtaining and modeling technical requirements as well as assuring consistency between requirements and other modeling constructs. To achieve this, a hierarchical interpretation of requirements is used. Additionally, the structure of active principles displays the system elements, their interfaces and interactions.

To add comfort to the analysis of multidisciplinary mechatronic systems, a concept of a Multifunctional Model Client (MMC) is developed (cf. [4]). The MMC serves three main purposes: It semi-automatically configures dynamic behavior models by combining models of the components for the specific problem. Also, it is linked to the system knowledge/system model and therefore able to ensure consistency between dynamic behavior models. And lastly, it provides access to reusable models (solution knowledge) that were made available via semantic web ontologies. Thus, the designer is able to analyze the integrated system by means of the assembled simulation model. For this purpose, MMC takes the desired level of detail into account and has all component models that are currently available at its disposal.

We use the technology of Semantic Web ontologies to formalize solution and system knowledge. The idea behind the Semantic Web is to enrich information with annotations in such a way as to enable machines as well as humans to understand and correlate the content [5]. Therefore, the knowledge of experts on the subjects involved is stored resp. modeled in so called ontologies. Ontologies provide a way to describe not only information on facts but also on the semantic relationships between concepts. Also, inference rules allow logical conclusions. In contrast to mere databases, so-called reasoners will then be able to expand the explicitly stored knowledge with implicitly existent knowledge that resulted from the logical relations and rules modeled. Thus, logical errors in knowledge representation become evident. The logical consistency of the ontology can be checked. To build up ontologies, we use the W3C standard OWL 2 [6]. Here, classes, datatype properties and object properties are subdivided and defined.

In order to integrate dynamic behavior models, to manage variability of their components and to achieve consistency, we use feature modeling techniques. These techniques provide an easy, understandable, and generic way of representing the variability information, independent of a specific application domain [7].

The remainder of the paper is as follows: At first we will explain the design process in general. Subsequently, we will focus aspects, which are important in this context. Thus, Section 3 deals with deriving and modeling technical requirements and the structure of active principles. It is pointed out how these two models can be used to reduce consistency issues. In Section 4 we then explain the concept of the Multifunctional Model Client, which is particularly useful in the discipline-spanning coordination. At last we will give a conclusion and an outlook for future work.

2. Design Process of Intelligent Dynamic Systems

As regards the design process, we rely on existing design methodologies, mainly the VDI guideline 2206 (cf. Figure 1) and the specification technique CONSENS [2,8], which divide the development process into three main phases: the discipline-spanning conceptual design or system design, the concurrent discipline-specific design and the system integration [8]. In all three of the phases modeling techniques, model analyses and other model-based methods facilitate the designers work. The process is highly iterative and one will have to return in previous phases repeatedly. In particular, synthesis and analysis steps will alternate very often. Nevertheless, the sequential process model constitute a process-based meta-model that provides an important guide for developers (see [1,9]).

The discipline-spanning conceptual design is carried out by a team of designers from the involved disciplines. The objective is to take the development order and initial requirements and create a conjoint understanding of the new product. The conceptual design phase results in the so-called principle solution, which is described holistically by a coherent system of partial models. An important partial model of CONSENS is the structure of active principles (or active structure), which displays the system elements and their interactions. Furthermore, idealized dynamic behavior models (DBM) come to use, which fit the discipline-spanning conceptual design phase, regarding their level of detail. With the help of these models, the functional capability of the principle solution is evaluated with respect to the dynamics. The designers can compare different principle solutions, configure the chosen configuration, and validate the dynamic function. Thus, the principle solution is tested against the specified requirements by means of idealized simulations. If the requirements can be met by the principle solution, the process continues and the system can be elaborated, otherwise failure is noticed in this early step. In combination, this ensures that discipline-specific engineering tasks can be carried out in parallel without major conflicts. The principle solution constitutes a prerequisite for seamless integration of the discipline-specific outcomes. The system knowledge, which is initially build up in the conceptual design and constantly expanded in the subsequent design steps, should be available to every domain expert.

The discipline-specific elaboration of the system is carried out concurrently by each engineering domain involved. The parameters of the previous idealized simulation serve as target values of the search for suitable products. These are called "solution knowledge", for which more detailed models are needed. These models e. g. describe the dynamic behavior or the specific shape (CAD) in detail and represent the product-specific characteristics. Manufacturers might provide both in the form of so-called solution elements to gain a competitive edge and open new distributive channels [10]. Using them, the various discipline-specific models and artifacts are elaborated and refined with the help of specialized methods and tools. Dynamic behavior models, for example, enable detailed controller design and verification of the integrated system-specific dynamic behavior. Model-based multi-domain analyses, as well as system optimization can take place under realistic conditions.

Development of the previously described systems requires early and ongoing tests of the characteristics, interdisciplinary key features and main functions. Integrating system simulations are particularly suited for this purpose. In addition to the original V-model we, therefore, explicitly added discipline-spanning coordination, in which 1) consistency is assured and 2) integrated simulations of the dynamic behavior are used to analyze whether the system still complies with the desired functionality of the stakeholders or not. Depending on the test case or the properties to be analyzed, the involved disciplines are integrated so that their interactions can be studied and optimized. In particular, we focus on integrating aspects that affect the system dynamics. Therefore, dynamic behavior models of the components have to be combined and configured in an appropriate level of detail, based on the actual objective. In the discipline-spanning coordination, dynamic behavior models of the components should represent the specific dynamic behavior on the system level and include boundaries and constraints. Abstraction from more detailed models are needed. In order to facilitate the designers work of handling and managing different models, solution and system knowledge, we developed the concept of a Multifunctional Model Client (MMC).

To illustrate our approach for an integrated model-based design process, we use the application example of two cooperating delta robots that juggle a ball by passing it to each other (see also [11]). The system comprises two identical, autonomous delta robots, which are equipped with a movable racket on their tool-center-points (see Figure 1). The rackets consist of tilt kinematics, three piezo force sensors and a racket plate. Because no optical sensors are used, the striking robot has to detect the point of impact using the three sensor signals, predict and communicate the ball flight trajectory by means of a dynamic model. On the one hand, this denotes an ambitious task for control engineers. Because of the limit information feedback, model-based

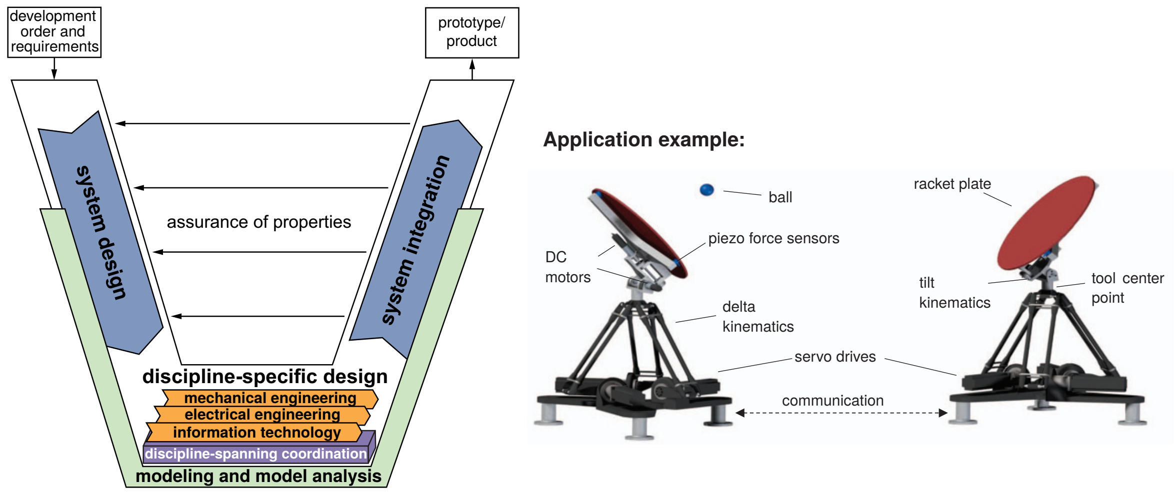

Fig. 1. V-Model of the development process (left, in dependence on [8]) and application example of cooperating delta-robots.

observers and frequent communication of the decentralized control units (e. g. for the gaming strategy or for the prognosis of the next strike) are needed. On the other hand, measuring the ball impact during the strike requires close collaboration of mechanical, electrical and control engineers. In order to meet the requirements of fast and reliable ball-detection, integration and integrated simulation of the discipline-specific results is essential. In summary, the cooperating delta robots constitute a representative of a new class of intelligent dynamic or cyber-physical systems that lead to new requirements on a systematic design process considering different disciplines. In the following, some important aspects of the described methodology to design intelligent dynamic systems like the cooperating delta robots will be explained exemplarily.

3. Modelling Technical Requirements and System Structure in a System Model

Several classifications of requirements have been presented. The most famous one is presented by Volere. Robertson and Robertson differentiate requirements into different categories, e.g. safety requirements or performance requirements [12]. The here used approach provides an assignment of different levels of requirements to a development process, and therefore serves a different aim. The levels give an orientation of what degree of detail is proper for a requirement used in a certain phase of development. Inside those levels, the common classifications (e.g. from Volere) can be maintained.

The discipline-spanning system model formalizes a conjoint view on the system. When the development progresses to discipline-specific design, inaccuracies from previous phases may lead to significant problems. It can be pointed out that it is crucial to assure consistency inside the development process [13]. A novel interpretation of requirements offers a valuable option to solve consistency issues in the transitions between development phases [14]. This interpretation is based on four hierarchical levels of requirements (Fig. 2). At the beginning of a development process, *goals* of a system exist. By a hierarchy of functions, functions can be identified, which fulfill the goals. To all defined functions, *function-oriented requirements* exist, which should be taken into consideration. When starting to develop a system architecture, different system elements evolve. Since these interact which each other, *domain-spanning requirements*, e. g. interfaces, need to be excerpted. Inside the domain-specific design phase, *domain-specific requirements* need to be obtained, which provide the desired degree of detail for requirements on the component level.

To ensure a consistent system development, sufficient methods support the requirements process. The CONSENS partial models can be extended to supply the developer with information about requirements. Particularly the hierarchy of functions and the structure of active principles are helpful to excerpt requirements. Beyond the extended partial models, the utilization of dynamical simulation models is a valuable possibility

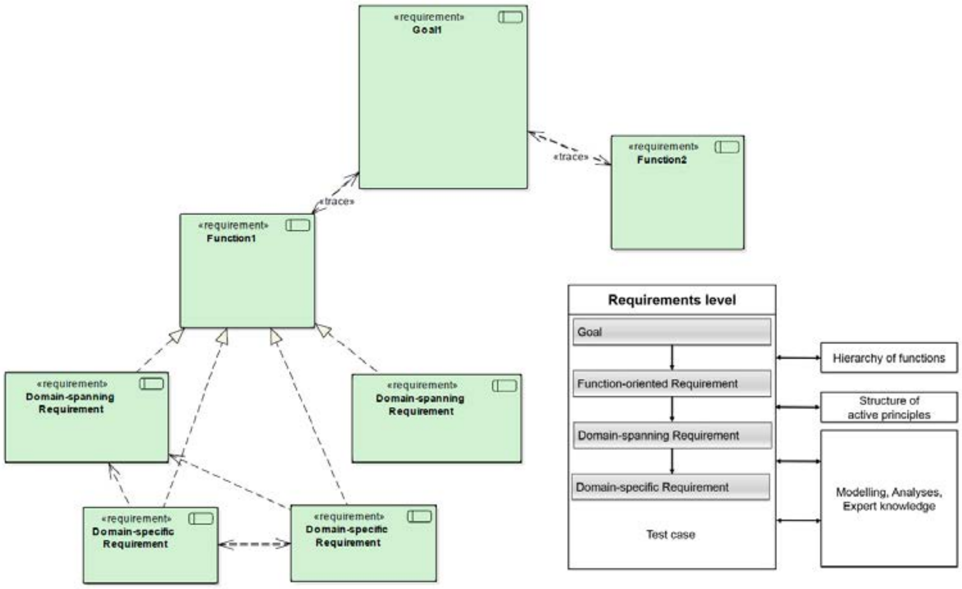

Fig. 2. Requirements Diagram with Process Overview

to obtain requirements. Unknown physical values can be reliably estimated. This way, step by step, the entire set of requirements evolves. To store the gained requirements, SysML offers the *requirements diagram* [15]. On the left side of Figure 2, the requirements diagram is shown in the hierarchical interpretation of the above mentioned four requirement levels. The associated supporting methods for the transitions can be seen on the right hand side.

A detailed description of the supporting methods can be found in [14]. For the development of the two cooperating delta robots, representatively for the utilization of partial models, the structure of active principles is shown in Figure 3. The structure of active principles displays the system elements, their interfaces and interactions. The connections between all system elements can be extracted. Corresponding to the requirements refinement process, these connections need to be added to the domain spanning requirements as system element interfaces. Since requirements are obliged to have certain features, e.g. verifiable [16], test cases should always be considered. Especially the development of large scale systems benefits from these systematic considerations.

For the application example, the structure of active principles (Fig. 3) discloses, e.g. the connection between the system elements *racket* and *deltaKinematics* to be a *Mechanical 3D*-connection, which constitutes an energy flow (*EnergyFlowSpecification*). That implies the exchange of translational, as well as rotational movement between the two system elements. A domain-spanning requirement of this interface can thus be excerpted. Subsequently, this leads to domain-specific requirements. Corresponding forces and momentums can be identified by model-based analyses. In general, the structure of active principles is equivalent to a first physically motivated model of the system, and therefore qualifies for further analyses.

4. Discipline-spanning Coordination with Multifunctional Model Client

To analyze the domain-specific dynamic behavior of the subsystems and their components, multiple dynamic behavior models are developed in different levels of detail and domains. However, in order to analyze the complex interactions and dependencies between them and to assure the required system characteristics (see Section 3), integrated models of the whole system are needed. These need to fit the varying modeling

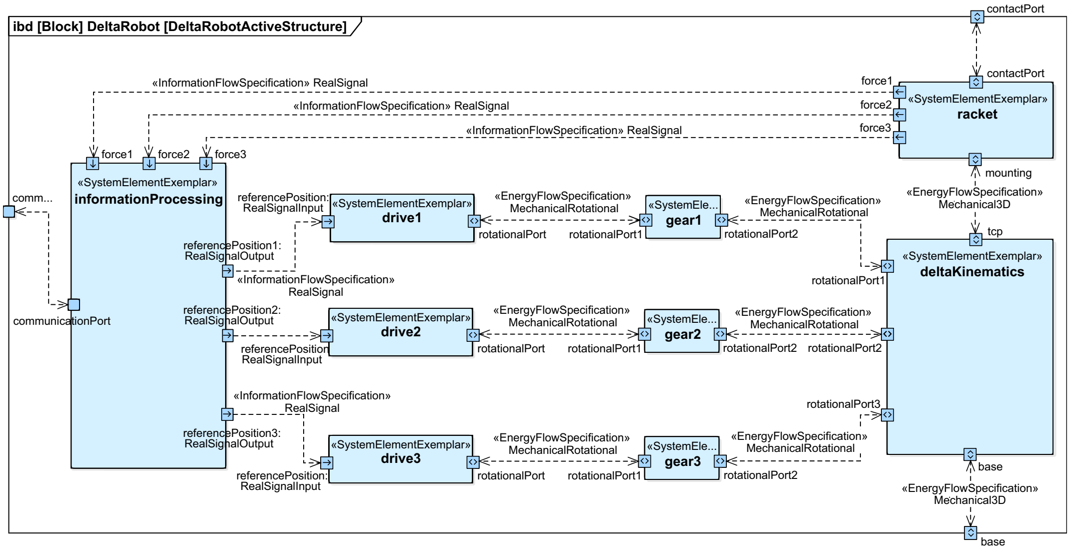

Fig. 3. Structure of active principles of a delta robot using the SysML4CONSENS profile [3]

objectives and analysis goals. Furthermore, consistency between the sub-models from different origins is crucial in order to get reliable simulation results. Certainly, the manual process of building up such models and maintaining consistency entails great effort. To reduce this, we developed the concept of a Multifunctional Model Client (MMC). The concept of Multifunctional Model Client (see Figure 4) adopts the present methodology of integrated model-based design. It is particularly useful in the course of discipline-spanning coordination. Here simulation models that analyze the dynamic behavior of a product need to be connected in order to analyze the integrated multidisciplinary dynamic behavior of the system. To fulfill the requirements, MMC accesses system knowledge formalized in a semantic system model. Furthermore, it enables efficient integration of approved solution knowledge, e. g. DBM that are prepared for reuse (cf. [17,18]). In summary, MMC has three main functions:

1. configure dynamic behavior models
   (e.g. combine components into DBM of the system using feature model)
2. ensure consistency
   (e.g. manage changes of parameters and interactions)
3. access reusable solution knowledge
   (e.g. enable reuse of DBM/make DBM available for reuse)

In the following we will explain these functions in more detail.

4.1. Function 1: Configure Integrated Dynamic Behavior Model

Modeling variability and commonality are the key elements in developing product families and product lines. The objective of the analysis of commonality and variability is to identify strategic reuse [19]. A feature model represents the information of all possible products of a software product line in terms of features and relationships among them [20]. In the presented approach, feature modeling is the base of the model client, which puts submodels together to create a simulation model for the entire system. The variants of levels of detail (modeling depths) and modeling tools of every system element of the mechatronic system are represented in such a variant model. We therefore rely on Lochbichler et al., who classified four levels of detail of dynamic behavior models [21]. They also more precisely defined and introduced the term modeling depth.

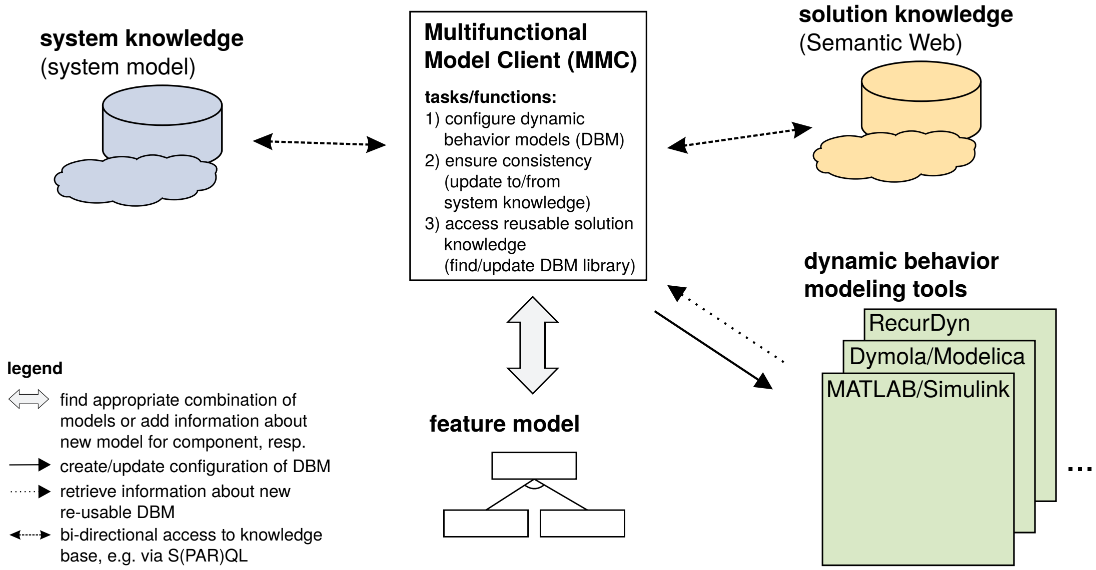

Fig. 4. Concept of Model Client (see also [4]).

In the conceptual design stage an active structure like the one in Figure 3 is defined. According to that, an initial feature model is created to offer the user the possibility to choose between different combinations of models. The MMC searches for an existing dynamic behavior models (DBM) of participant system elements in the available model libraries, which are represented in either the system or solution knowledge base.

An excerpt of the feature model of the delta robot exported from the active structure is demonstrated in Figure 5. This feature model is used by MMC in order to give the user the opportunity to choose a possible combination of models from a choice tree, as shown in Figure 5 (left hand side). Based on the integrated simulation, the user is then able to verify the configuration of racket and delta kinematics. The dependency between the racket and delta kinematics models are shown as a constraint that reflects the connection of system elements in the active structure. The desired integrated model is built up by mapping the necessary interfaces based on the valid configuration chosen. A configuration is valid, if the combination of features is allowed by the feature model (i. e. it fulfills the semantics of groups and all cross-tree constraints) [22]. Similarly, different modeling depths of these components can be selected.

According to the choice in Figure 5, one section of the physical model can be created in a signal flow tool and the part of the multibody dynamics in another specialized simulation tool. In order to join them, an appropriate connection is required, because of structural differences in the tools and the models. Hence, all relevant properties respectively inputs, outputs, ports and parameters of a component are illustrated with type and unit in the feature model. By processing information in the feature model, it is determined which adapters and unit converters are required for the interfaces in order to connect the submodels. For this purpose the MMC searches in the table of adapters and figures to find out, how a given interface can communicate with other elements. If an appropriate adapter is found in the library, it is added to the model. Otherwise the interfaces of missing adapters are displayed as a result. The MMC also performs a check on the units between interfaces. In the case of a unit inconsistency, the unit table is searched to find matching SI units and recalculation may be executed [4].

4.2. Function 2: Ensure Consistency with System Model / System Knowledge Base

While consistency of the different dynamic behavior models can be managed using a feature model, reflecting of all information from system knowledge, like linked requirements for example, needs to be established. We

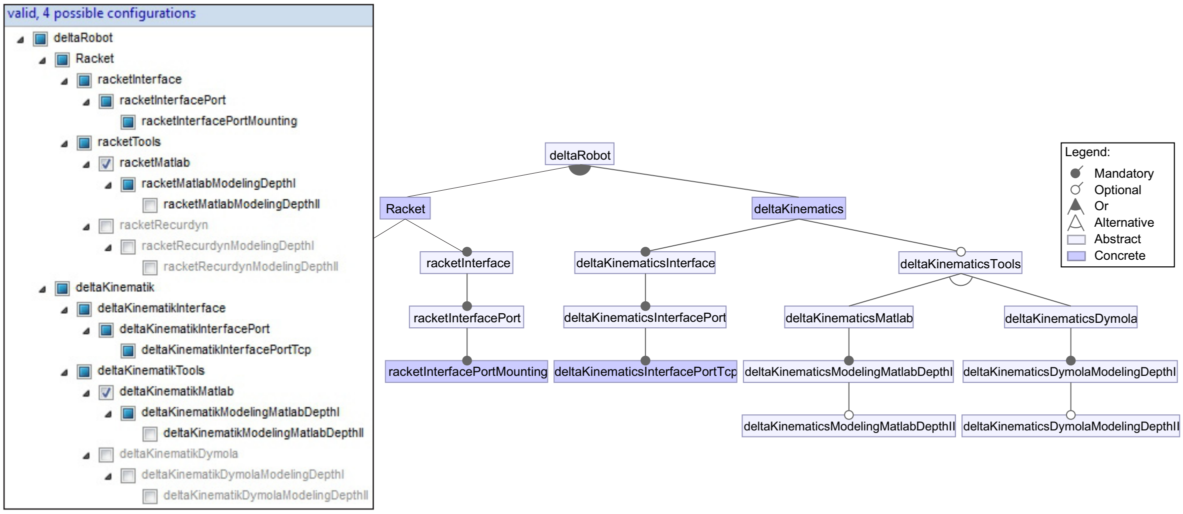

Fig. 5. Excerpt of feature model based on active structure (right), valid configuration (left)

therefore aim at using a knowledge model of the system that constitutes a central, semantically defined image of the system and can be used as a reliable source of information. This so-called system knowledge model in the form of ontologies gathers information on the development results of the participating disciplines, makes explicit and implicit system knowledge available and shows dependencies. It is continuously maintained during the development. Thereby, the communication between disciplines and possibly even with suppliers can be improved. This is particularly useful in the discipline-specific elaboration and the discipline-spanning coordination phase and extends scope and expressiveness of previous system models, which are modeled e. g. with SysML. Furthermore, it helps ensuring consistency between the specific models and development tools of the different disciplines. The semantic system model comprises information on three main aspects:

- Characteristics: It gathers characteristics/parameters of the system and system-relevant features of system elements. This includes both quantitative and textually described characteristics as well as links to other development artifacts.
- Model representations: Information on existing discipline-specific and interdisciplinary models are modeled.
- Goals: To track and assure required properties, it is necessary that the specified targets, like the system requirements and functions are be mapped and linked.

By connecting to the system knowledge model via MMC for bi-directional information exchange, both the semantic system model and the simulation models are kept up to date.

Figure 6 shows a severely reduced excerpt of the constructed ontologies, which can be divided in two parts. The terminology constructs and more universal contents in the Terminological Box (T-Box) serve as a meta-model ("language") for the formulation of specific statements and facts in Assertional Box (A-Box) (see [23]). Accordingly, the T-Box mainly includes higher-level concepts and rules, while the A-Box primarily contains concrete factual knowledge of individuals/instances. In order to be able to model the semantics of the cooperating delta robots, respective classes for *System Elements*, *Requirements*, *Functions*, *Engineering Models* etc. are introduced in the T-Box. It also contains properties to describes relations and characteristics. Object-Properties like for example *hasModel* can be used in the semantic system model (A-Box) to link a system element with a certain model representing it. On the other hand, Datatype-Properties assign literal values to an individual or class. In Figure 6 this is exemplarily displayed for the system element *delta kinematics*, which is connected with two different discipline-specific model representations: an idealized dynamic model (*DeltaIdealizedDynamics*) and a shape model *Delta3DShapeModel*, which is used by mechanical engineers. Furthermore, the modularization, which was introduced with the structure of active principles (Figure 3), is reflected by the *comprisesSystemElement* property. While many information of respective system elements can be stored by literal values like e. g. the mass, the dimensions or an ID, they are also used to make quantifiable knowledge of assigned technical requirements available. To illustrate this, Figure 6 picks up the example of the racket plate's damping (cf. Section 3). Also, the information on the interfaces and connections of the system elements are modeled (not shown).

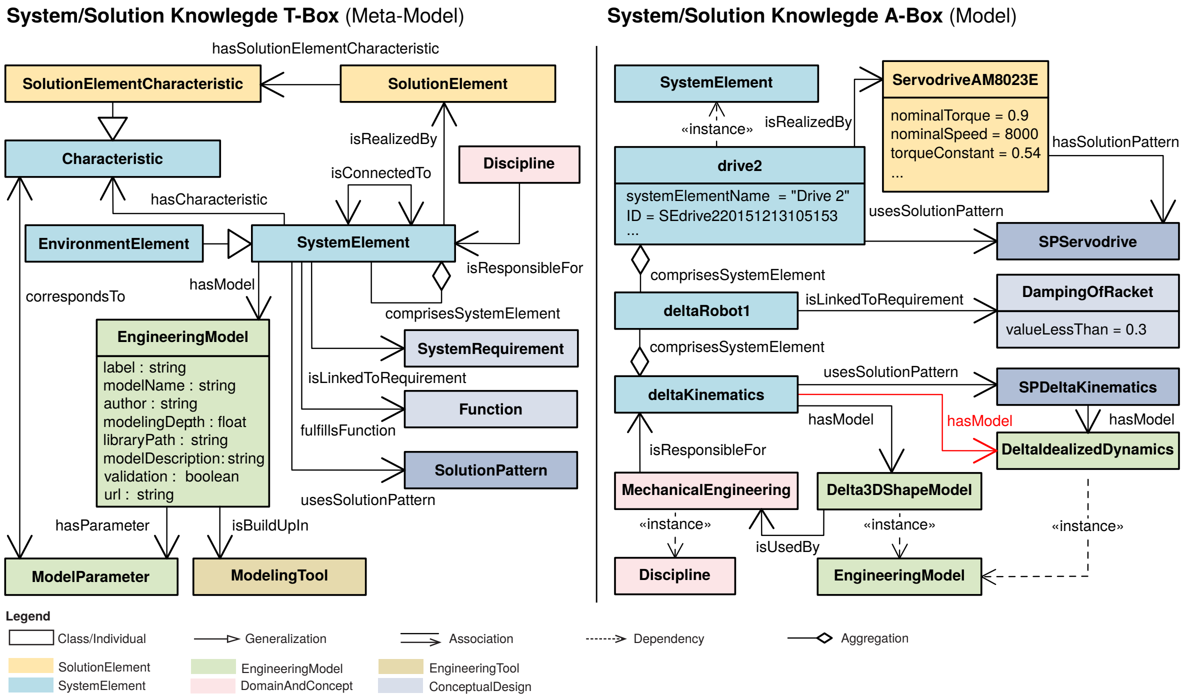

Fig. 6. Excerpt of ontologies to describe system/solution knowledge (illustrated in a UML class diagram).

4.3. Function 3: Utilize Solution Knowledge

Using approved solution knowledge addresses two main obstacles in mechatronic design process [10,24]. On the one hand, there is the modeling effort, which is a big barrier to more extensive model-based design. This can be overcome by reusing and combining ready-to-use submodels. But, preconditions have to be met in order to make it efficient (cf. [25]). Reusable DBM have to be documented in a way that their purpose and underlying assumptions become obvious. Modeling engineers need to make sure that provided models are not being used in conditions where they give incorrect results. Models should rather be tailored to fit a certain step (e.g. conceptual design) of the design process. Additionally, search for suitable existing DBM has to be supported by tools. Therefore, meta-information about them must be available in a solution knowledge base. This includes information about parameters and the modeling depth index [21]. MMC assists engineers by 1) enabling (semantic) search for reusable DBM that are appropriate and 2) testing and automatically retrieving meta-information from models that are to be stored in the knowledge base.

On the other hand, mechatronic products are highly complex and error-prone systems that ought to be developed in decreasing time-to-market cycles. In this situation, reusing complete and approved solution elements, like the delta-kinematics or the servo drives, can help the developers. However, they often find it difficult to discover the most sufficient solution element for components of the system. This is due to the enormous variety of commercially available products, which are presented in varying, company-specific terminologies and classifications. It was therefore suggested that suppliers provide information about their so-called solution elements via the Semantic Web in the future [10]. Thus, MMC offers a possibility for developers to gain bi-directional access to detailed solution knowledge.

Figure 6 gives an impression of how solution knowledge and its connection to system knowledge is modeled (see also [10,24]). Here the specific servo drive *AM8023E* is represented by an individual of the class *SolutionElement*. Assume that it was chosen to use a servo drive to move the delta kinematics. So, in the conceptual stage of the design of the cooperating delta robots the corresponding system element individuals were connected with the solution pattern *SPServodrive* in the semantic system model. However, the fact that *AM8023E* is a solution element of it is modeled in the solution knowledge ontologies. Now parameters and results of the simulation in the concept phase serve as inputs for the semantic search. They filter the great amount of drives using the modeled solution elements characteristics. Thus, the specific servo drive *AM8023E* can be found as a suitable solution element. The respective models and aspects of it can be used and inserted. In the semantic system model the *isRealized* property is set. As a results all information on this drive are also available in the semantic system model and to the designers. The *hasModel* property between the delta kinematics system element and the idealized dynamic behavior model *DeltaIdealizedDynamics* (marked in red) exemplarily illustrates knowledge, which is implicitly available. It can be inferred by the reasoning algorithm, because the corresponding solution pattern *SPDeltaKinematics* is used.

5. Conclusion and Outlook

In this paper, an approach for the integrated model-based design of intelligent dynamic systems is presented. As a prerequisite to solve consistency issues a hierarchical interpretation of detailed technical requirements is introduced and a discipline spanning coordination phase is added to the original V-model of the VDI guideline 2206. Also the concept of a Multifunctional Model Client (MMC) is introduced, which combines outcomes of the different disciplines to analyze mechatronic systems. The MMC is created for combining and configuring of dynamic behavior models of the components in an appropriate level of detail. Through the adopting of feature modeling approach, consistency between different disciplines is assured. The MMC supports sharing of common information between development team members, because bi-directional information exchange to system and solution knowledge is established. Therefore, both information on the system and information on suitable solution elements is modeled in Semantic Web ontologies. In summary, MMC helps developers to follow an integrated model-based design process of mechatronic systems step by step.

In future work, all aspects of MMC will be implemented in a piece of software. To achieve seamless integration of the presented approach in everyday work, we will focus on coupling it with widely used tools like Enterprise Architect, MATLAB/Simulink or Dymola. Therefore, the knowledge bases, including solution elements and reuseable models, as well as adapters and unit converters for matching interfaces, will be enlarged. However, as the representation of system knowledge and solution knowledge comply, every successful development possibly supplies new solution elements. Moreover, we plan to evaluate the presented concept with more examples.

Acknowledgments

This research and development project is funded by the German Federal Ministry of Education and Research (BMBF) within the Leading-Edge Cluster "Intelligent Technical Systems Ost-WestfalenLippe" (it's OWL) and managed by the Project Management Agency Karlsruhe (PTKA). The authors are responsible for the contents of this publication.

References

[1] Alvarez Cabrera, A., Foeken, M., Tekin, O., Woestenenk, K., Erden, M., De Schutter, B., et al. Towards automation of control software: A review of challenges in mechatronic design. Mechatronics 2010;20(8):876–886.

[2] Gausemeier, J., Lanza, G., Lindemann, U.. Produkte und Produktionssysteme integrativ konzipieren: Modellbildung und Analyse in der frühen Phase der Produktentstehung. Carl Hanser Verlag; 2012.

[3] Kaiser, L., Dumitrescu, R., Holtmann, J., Meyer, M.. Automatic verification of modeling rules in systems engineering for mechatronic systems. In: Proceedings of the ASME IDETC/CIE 2013. Portland, Oregon, USA; 2013, p. V02BT02A009.

[4] Abrishamchian, F., Oestersötebier, F., Trächtler, A.. Feature model approach for managing variability of dynamic behavior models in mechatronic systems. In: Proceedings of the ASME 2015 International Mechanical Engineering Congress & Exposition. Houston, Texas: ASME; 2015,.

[5] Berners-Lee, T., Hendler, J., Lassila, O.. The semantic web. Scientific American 2001;284(5):34–43. URL: `http://www.sciam.com/article.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21`.

[6] Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P.F., Rudolph, S.. Owl 2 web ontology language - primer (second edition). W3C; 2012. URL: `http://www.w3.org/TR/owl2-primer/`.

[7] The Eclipse Foundation, . EMF feature model, eclipsepedia. Online; 2014. URL: `https://wiki.eclipse.org/EMF_Feature_Model`; https://wiki.eclipse.org/EMF_Feature_Model accessed 15-October-2014.

[8] Verein Deutscher Ingenieure (VDI), . VDI guideline 2206: Design methodology for mechatronic systems. Beuth Verlag 2004;.

[9] Strahringer, S.. Ein sprachbasierter Metamodellbegriff und seine Verallgemeinerung durch das Konzept des Metaisierungsprinzips. In: Pohl, K., Schürr, A., Vossen, G., editors. Modellierung; vol. 9 of *CEUR Workshop Proceedings*. CEUR-WS.org; 1998,.

[10] Oestersötebier, F., Just, V., Trächtler, A., Bauer, F., Dziwok, S.. Model-based design of mechatronic systems by means of semantic web ontologies and reusable solution elements. In: Proc. of the ASME 2012 11th Biennial Conference on Engineering Systems Design and Analysis. 2012, p. 647–656.

[11] Shareef, Z., Just, V., Teichrieb, H., Trächtler, A.. Design and control of cooperative ball juggling delta robots without visual guidance. Robotica 2015;:1–17.

[12] Robertson, S., Robertson, J.. Mastering the requirements process: Getting requirements right. 3rd ed. ed.; Upper Saddle River, N.J: Addison-Wesley; 2013. ISBN 978-0321815743. URL: `http://proquest.tech.safaribooksonline.de/9780132942850`.

[13] Lankeit, C., Lochbichler, M., Trächtler, A.. Challenges in requirements engineering for mechatronic systems ? problem analysis and first approach. Jurnal Teknologi 2015;76(4).

[14] Lankeit, C., Lochbichler, M., Oestersötebier, F., Trächtler, A., Landwehr, M.. From goals to systems via technical requirements and physical models - design of an intelligent cash handling system. In: Proceedings. IEEE International Symposium on Systems Engineering; IEEE; 2015,.

[15] Weilkiens, T.. Systems engineering with SysMLUML: Modeling, analysis, design. Amsterdam: Morgan Kaufmann OMG Press/Elsevier; 2007. ISBN 978-0123742742.

[16] Systems and software engineering – life cycle processes –requirements engineering. ISO/IEC/IEEE 29148:2011(E) 2011;:1–94doi:10.1109/IEEESTD.2011.6146379.

[17] Mocko, G., Malak, R.J., Paredis, C.J., Peak, R.. A knowledge repository for behavioral models in engineering design. In: Proc. ASME Design Engineering Technical Conf., Computers and Information in Engineering Conf. 2004,.

[18] Paredis, C., Diaz-Calderon, A., Sinha, R., Khosla, P.. Composable models for simulation-based design. Engineering with Computers 2001;17:112–128.

[19] Clauß, M., Intershop, J.. Modeling variability with UML. In: GCSE 2001 Young Researchers Workshop. Citeseer; 2001,.

[20] Sayyad, A., Ammar, H., Menzies, T.. Software feature model recommendations using data mining. In: Recommendation Systems for Software Engineering (RSSE), 2012 Third International Workshop on. 2012, p. 47–51.

[21] Lochbichler, M., Oestersötebier, F., Trächtler, A.. Dynamic behavior models and their modeling depth in the design process of mechatronic systems. In: Proc. of the ASME 2014 International Mechanical Engineering Congress & Exposition IMECE 2014. Montreal, Québec, Canada: ASME; 2014,.

[22] Thüm, T., Kästner, C., Benduhn, F., Meinicke, J., Saake, G., Leich, T.. Featureide: An extensible framework for feature-oriented software development. Science of Computer Programming 2014;79:70–85.

[23] Hitzler, P., Krötzsch, M., Rudolph, S., Sure, Y.. Semantic Web: Grundlagen. Berlin: Springer; 2008.

[24] Oestersötebier, F., Dziwok, S., Bauer, F., Trächtler, A., Schäfer, W., Gausemeier, J.. Unterstützung des mechatronischen Entwurfs durch die effektive Suche nach Lösungselementen mithilfe von semantischen Technologien. In: Proc. of Tag des Systems Engineering 2012. 2012,.

[25] Robinson, S., Nance, R.E., Paul, R.J., Pidd, M., Taylor, S.J.. Simulation model reuse: definitions, benefits and obstacles. Simulation modelling practice and theory 2004;12(7):479–494.